Report on the standardization project "Formal methods in conformance testing"

Dieter Hogrefe, Stefan Heymer
Institut für Telematik

Universität Lübeck
D-23538 Lübeck
{hogrefe,heymer}@itm.mu-luebeck.de

Jan Tretmans
Tele-Informatics and Open Systems group
Dept. of Computer Science
University of Twente
NL-7500 AE Enschede
tretmans@cs.utwente.nl

Abstract

This paper presents the latest developments in the "Formal Methods in Conformance Testing" (FMCT) project of ISO and ITU–T. The project has been initiated to study the role of formal description techniques in the conformance testing process. The goal is to develop a standard that defines the meaning of conformance in the context of formal description techniques. We give an account of the current status of FMCT in the standardization process as well as an overview of the technical status of the proposed standard. Moreover, we indicate some of its strong and weak points, and we give some directions for future work on FMCT.

Keywords

Conformance testing, formal methods, standardization.

1 INTRODUCTION

Formal Methods in Conformance Testing (FMCT) is a project initialized by ITU–T and ISO to study the meaning of conformance in the context of formal description techniques such as SDL, LOTOS, and Estelle. What does it mean that a product conforms to a specification given in such a formal notation? The first status report of FMCT has been given in [Hog95]. Please consult [Hog95] for a general introduction to the subject. FMCT is coming to an end now, and it is time to review the current developments.

Given a formal specification, how can one find out whether an implementation conforms to this specification?

The motivation of this activity is the existence of formally described standards and therefore the need to define the meaning of conformance with respect to formal specifications. As the use of formal methods in conformance testing is a very general problem in software engineering, its principles should be applicable to any application area, in particular any application area in telecommunications, e.g., OSI, ODP and IBCN.

Since the Conformance Testing Methodology and Framework (CTMF) [ISO91] defines conformance in a very general, informal sense, it serves as a natural basis for the more formal work on FMCT. By nature there is a close relationship between CTMF and FMCT. In particular, the concepts and their definitions should be compatible between the two documents.

The FMCT project is currently maintaining two working documents:

- Framework for FMCT (CD 13245–1, [ISO96])
- Guidelines for FMCT (CD 13245–2, [ISO95])

The framework gives some general information about FMCT and defines the basic concepts and terminology. Annex A, which will be incorporated into the standard, but which will be informative, explains the relationship between the FMCT concepts and the existing FDTs Estelle, LOTOS and SDL. The annex explains, for example, the concept of the implementation relation with respect to the three languages and gives an interpretation of PICS proforma and PICS. The annex will also contain a tutorial.

The guidelines, which will be published separately, deal with test generation methods as they have been developed for finite state machines and other models. A common example, the INRES protocol and service [Hog92], is used to show the applicability of existing test generation methods to an OSI protocol specification. INRES is not a real protocol, but it contains many features of OSI protocols. It is an abridged version of the Abracadabra protocol used in [ISO90] for illustration.

2 CURRENT STATUS

The work on FMCT is done in close cooperation between ITU-T and ISO/IEC. The cooperation was achieved by running a collaborative team. This cooperation proved to be very fruitful and gathered resources from both ITU–T and ISO in order to progress a single project. The "Framework for FMCT" underwent a ballot within ISO until mid of April 1996.

The results of this ballot were forwarded to an editing meeting for the ISO committee draft scheduled for mid May 1996 in Kansas City. The results are the following:

- Out of 23 national bodies that were members of SC21, 11 bodies voted.
- Of these 11, there were the following votes:
  - Abstain: 2
  - In Favour: 8
  - Against: 1

The main causes for the "against" vote (other than typographical errors) were uncertainty about the practical relevance of the proposed standard (cf. section 4 on the weakness of FMCT), and that some of the necessary preconditions for the formal treatment of conformance testing seemed to be dubious. On an editing meeting inside ITU–T in Geneva in April 1996 it was noted, that a tutorial on the use of FMCT would help this. Annex A of the FMCT document could be extended in such a way. This could also help in terms of a wider acceptance of this work.

It was also noted on the same meeting that the relationship between CTMF and FMCT has changed over the study period from formalizing CTMF to a more independent point. Many points in FMCT are not related to CTMF anymore.

Once the comments given with the "against" vote are disposed (i. e. resolved), the Committee Draft on FMCT (document number CD 13245–1) shall be accepted by ISO. It will then be put on ballot as a Draft International Standard (DIS). In the mean time the document will put forward as draft recommendation Z.500 to ITU–T. The current project planning aims at a recommendation and international standard, respectively, in 1998.

The project planning for the "Guidelines for FMCT" is shifted by one year with respect to the framework.

Beyond the work done during the study period from 1993 to 1996, work on verification and testing other than conformance testing is desirable within the continuation of the FMCT project.

The editing meeting in Geneva in March 1996 discussed intensively the issue of interoperability testing. It was noted that interoperability testing seems to fit into the FMCT framework. This can be seen in figures 1 and 2, where figure 1 shows the situation encountered in conformance testing, while figure 2 shows the situation for interoperability testing. In both figures IUT is the implementation under test, UT is an upper tester, and LT a lower tester. For interoperability testing additionally the specification $Spec$ is taken into account. In both figures the dashed lines enclose the part of the combined specifications for which test cases could be generated.

3 TECHNICAL STATUS OF FMCT

The FMCT framework is presented at a high level of abstraction, e.g., it abstracts from specific test generation algorithms, even from a specific formal description technique. The framework defines terminology, abstract concepts, and minimal requirements on, and relations between these concepts. The main ingredients of the framework are concerned with how to define conformance of an implementation with respect to a formal specification [ISO96, section 6], how to reason formally about testing concepts, such test architecture and test execution [ISO96, section 7], and how to use testing concepts to test for conformance [ISO96, section 8]. These three concepts are briefly described, without being complete; a complete description can, of course, be found in the standard itself.

Conformance Conformance concerns the definition of what is a correct implementation of a formal specification. FMCT only defines what is necessary to define conformance in a

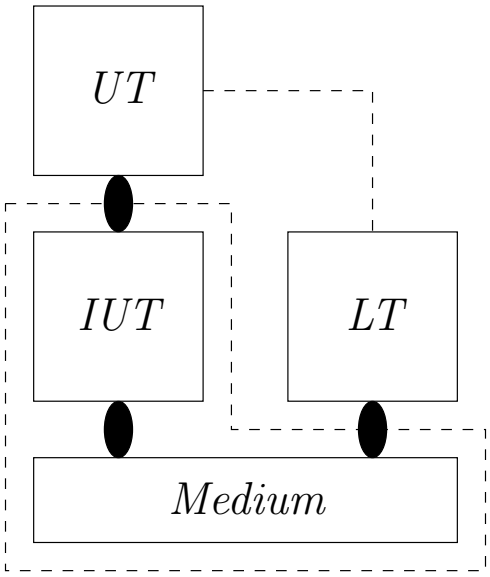

Figure 1: Test architecture for conformance testing

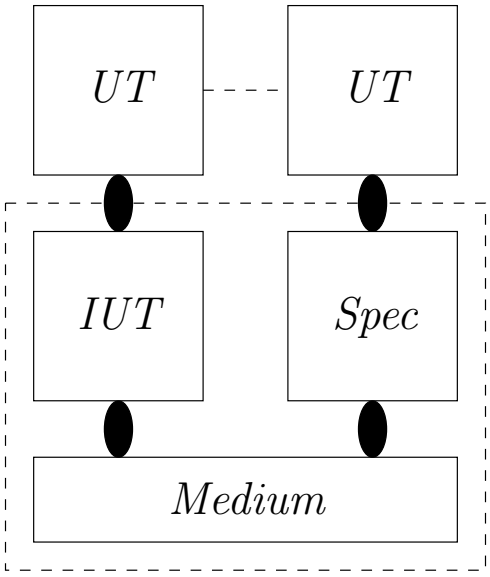

Figure 2: Test architecture for interoperability testing

formal setting, not how conformance is defined in a particular case. To define conformance it is assumed that there is a formal specification of required behaviour, $s \in SPECS$, where $SPECS$ denotes the formal description technique at hand, and a concrete implementation under test $IUT \in IMPS$, where $IMPS$ is the universe of all possible implementations. In order to reason formally about the concrete, physical implementations, it is assumed that each implementation can be modelled by a formal object m_{IUT} in a formalism $MODS$, which is referred to as the universe of models. This hypothesis is referred to as the test assumption. Conformance shall then be expressed by an implementation relation between formal models of implementations and specifications: $\mathbf{imp} \subseteq MODS \times SPECS$.

Testing Testing consists of performing experiments, specified in test cases, against an implementation under test, and observing its responses. Testing occurs in a test architecture, which is an abstract description of the environment in which the IUT is tested. It describes the relevant aspects of how the IUT is embedded in other systems during the testing process, and how the IUT interacts with these systems. The embedding systems are called the test context, and it is formalized as a function $\mathcal{C} : MODS \rightarrow MODS$ mapping behaviour of the IUT to behaviour of the IUT in its test context: $\mathcal{C}(IUT)$.

Test cases are specified in a test notation, referred to as $TESTS$, and a set of them is called a test suite. The process of running a test against a concrete implementation is called test execution. Test execution leads to an observation in a domain of observations OBS. Each observation is interpreted by assigning a verdict to it: $verd_t : OBS \rightarrow \{\mathbf{pass}, \mathbf{fail}\}$. An $IUT \in IMPS$ **passes** a test suite $T \subseteq TESTS$ if test execution of all its test cases leads to an observation with verdict **pass**.

A function $exec : TESTS \times MODS \rightarrow OBS$ is required, which models test case execution, i.e., given a test case $t \in TESTS$ and a model of an implementation $m \in MODS$, $exec(t, m)$ calculates the observation in OBS that results from executing t with the model m. By relating the observations made during the real test execution of the IUT with the tester to the observations calculated with $exec$, it can be concluded whether the model of IUT is in the subset of models for which a **pass**-verdict is calculated. This subset is called the formal test purpose, and it is denoted as P_t:

$$\begin{array}{ll} \text{let} & P_t \;=_{def}\; \{\; m \in MODS \mid verd_t(exec(t, m)) = \mathbf{pass} \;\} \\ \text{then} & IUT\ \mathbf{passes}\ t \iff m_{IUT} \in P_t \end{array} \tag{1}$$

and moreover for a test suite T:

$$\begin{array}{ll} \text{let} & P_T \;=_{def}\; \bigcap_{t \in T} P_t \\ \text{then} & IUT\ \mathbf{passes}\ T \iff m_{IUT} \in P_T \end{array} \tag{2}$$

Conformance testing In conformance testing the notions of conformance, by means of the implementation relation **imp**, and of testing are linked. This means that a test-suite generation algorithm is required, such that each possible implementation IUT passes the generated test suite (if and only) if it is conforming. Such a test-suite generation algorithm is expressed as a function $gen_{\mathbf{imp}} : SPECS \rightarrow \mathcal{P}(TESTS)$; ($\mathcal{P}(TESTS)$ is the powerset

of *TESTS*, i.e., the set of sets of test cases, so the set of test suites):

$$m_{IUT} \textbf{ imp } s \quad \begin{array}{c} \overset{sound}{\Longrightarrow} \\ \overset{exhaustive}{\Longleftarrow} \end{array} \quad IUT \textbf{ passes } gen_{\textbf{imp}}(s) \tag{3}$$

Since testing usually involves only searching for errors (the left-to-right implication in (3)), while it is never certain that all possible errors have been found (the right-to-left implication), FMCT requires that all generated test suites shall be sound.

To quantify the error-detecting capability of a sound test suite a fault coverage measure can be defined, which expresses the extent to which a sound test suite is exhaustive:

$$cov: \mathcal{P}(TESTS) \longrightarrow [0,1] \quad \text{satisfying} \quad P_{T_1} \supseteq P_{T_2} \text{ implies } cov(T_1) \leq cov(T_2) \tag{4}$$

Moreover, a cost function can be specified to define the costs of generating, implementing, and executing a test suite:

$$cost: \mathcal{P}(TESTS) \longrightarrow \mathbf{R}_{\geq 0} \quad \text{satisfying} \quad T_1 \subseteq T_2 \text{ implies } cost(T_1) \leq cost(T_2) \tag{5}$$

Apart from the items mentioned above, FMCT distinguishes between parameterized and instantiated specifications, it deals with implementation options, implementation conformance statement (ICS), and static conformance requirements, which state requirements on the sets allowable implementation options (the conformance aspects defined by means of an implementation relation are referred to as dynamic conformance). Moreover, FMCT defines fault models, and it presents several test-suite size reduction strategies: strategies to reduce the size of a test suite while preserving soundness. For details we refer to the (emerging) standard itself [ISO96].

4 STRENGTH, WEAKNESS, AND FUTURE WORK

The high level of abstractness of the framework for FMCT is its strength, but at the same time it is also its weakness. Making use of this abstract framework requires instantiating its concepts with specific choices for the formal description technique (*SPECS*), for the set of models (*MODS*), for the implementation relation (**imp**), for test generation algorithms, etc. Requirements for such an instantiation are defined in [ISO96, section 9]. The abstractness allows for very many different instantiations, making the standard applicable in many different areas, and making it easily adaptable to the specific needs of a particular testing problem. It also makes that it can cope easily with changes in technology and with evolving testing methods, since FMCT is not restricted to a particular specification technique, test generation method, or test-suite quality criterion, expressed as a coverage measure.

On the other hand, the abstractness and openness also mean that FMCT is not of direct help to daily testing problems. For any application of the testing framework it first needs to be instantiated, and this is not done in the present FMCT document. Annex A of FMCT presents possible instantiations for the standardized formal description techniques

Estelle, LOTOS, and SDL, but these instantiations are more intended to be illustrative than of practical value.

Hence, the most important next step in the standardization effort for formal protocol conformance testing is filling in the abstract framework with concrete formal description techniques, implementation relations, test contexts, test generation algorithms, test-suite size reduction techniques, coverage measures, etc. Some work in this direction has already been done: the document of Guidelines for FMCT [ISO95] contains an list of current, state-of-the art techniques in protocol conformance testing, and how they could fit within the framework. This section will discuss some issues which arise when instantiating the framework with concrete implementation relations, etc., thus giving some directions for necessary future work on FMCT. The discussion is structured according to the three main concepts discussed in section 3: conformance, testing concepts, and conformance testing.

Conformance To define conformance by means of an implementation relation, first the specification language *SPECS* and the class of models *MODS* need to be determined. Actually, the precise language is less important than the semantic model on which it is based, as is also apparent from Annex A of FMCT: although many different formal languages exist, the ones that are of interest for protocol conformance testing can usually be expressed semantically in some sort of state machine. Nowadays, mainly two families of state machines prevail: Finite State Machines (FSM) (or Mealy Machines) and Labelled Transition Systems (LTS). The main difference between them is illustrated in figure 3: while an FSM has transitions labelled with pairs of (input,output), an LTS specifies interactions, which can, but need not be interpreted as input or as output. An FSM thus implicitly makes the assumption of synchronicity between input and output: every input or stimulus immediately results in an output or response (possibly the empty output), while in LTS each input or output has its own transition.

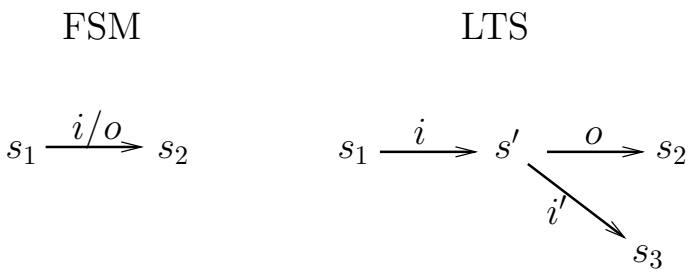

Figure 3: FSM and LTS transitions

Implementation relations for FSM, i.e., $SPECS = MODS = FSM$ and $\mathbf{imp} \subseteq FSM \times FSM$, are usually based on a some kind of relation between the states of the implementation and the states of the specification (isomorphism or homomorphism), where mostly the assumption is made that the number of states in the FSM implementation model is not larger than the number of states in the FSM-specification.

For LTS, i.e., $SPECS = MODS = LTS$ and $\mathbf{imp} \subseteq LTS \times LTS$, many possible implementation relations can be found in the literature, e.g., bisimulation equivalence, trace

preorder/equivalence, testing preorder/equivalence, conformance **conf** etc. [Gla93]. Consensus exists that an implementation relation suitable for conformance testing, should be based on a reasonable notion of observation, i.e., it should be possible to detect a non-conforming implementation with a test and a 'reasonable' observation; an implementation relation should not require properties on an implementation which can never be tested with a 'reasonable' testing scenario. Since many of the above mentioned implementation relations were not developed with conformance testing in mind, but applications like verification, model checking, and theorem proving, not all of them are suitable for conformance testing, e.g., bisimulation equivalence is not testable with a realistic testing scenario [Abr87]. Although consensus consists about the non-suitability of bisimulation equivalence, no consensus exists yet about what should be the ideal implementation relation.

A property of the above mentioned relations on LTS is that they do not distinguish between inputs and outputs. Instead, they use abstract interactions for communication, where it is assumed that two communicating partners negotiate to establish an interaction which they will jointly execute. It is questionable whether such a communication scheme coincides with observations made of realistic systems, which usually do distinguish between inputs and outputs, as FSM models do. To overcome this problem, lately implementation relations have been considered for LTS that do distinguish between inputs and outputs, and where input actions are always enabled in implementations, such as the relations quiescent preorder [Vaa91], $R_1, ..., R_5$ [Pha94], and **ioconf** [Tre95].

Consensus exists that a (very) limited set of suitable implementation relations should be defined as a prerequisite for further development of FMCT, however, no consensus exists yet, which these implementation relations are.

Testing Like for *SPECS* and *MODS*, a choice must be made for the test notation *TESTS*. Many languages could be used, the best-known of which is TTCN [ISO91, part 3], however, lack of a completely specified formal semantics for TTCN hinders its application in a formal context. Extensions and modifications have been proposed [FH95, WP96], but they lack the general acceptance and tool support of TTCN. Either TTCN should be better defined, or a reasonable alternative should be proposed.

In defining its general testing concepts, FMCT makes certain implicit assumptions. One of these is that an observation corresponding to the calculated observation $exec(t, m)$ can always be made. This might pose problems, especially when test execution consists of multiple test runs, and when due to nondeterminism it is never certain whether all possible test runs have really been obtained. The current FMCT framework does not have the possibilities to deal with such uncertainties; more on this can be found in [HT96].

Conformance testing For conformance testing we need to generate some test cases. Again, FMCT is very general and abstract by considering test generation as a function from *SPECS* to $\mathcal{P}(TESTS)$ with the only requirement that the produced test suites shall be sound. No clue is given how to obtain such a test generation algorithm, or how to show its soundness.

Depending on the specification formalism and implementation relation many test generation algorithms have been presented in literature, see also the Guidelines document.

unfortunately, not all of them have been shown to be sound. Most test generation algorithms have been developed for FSM: Transition Tours, W-method, Distinguishing Sequences, Characterizing Sequences, Unique Input-Output Sequences (UIO) with some variants (UIOv, UIOg, ...), etc. Mostly they are restricted to work with deterministic FSM, and they can be proved to be sound and exhaustive if the additional assumption is made that the number of states in the IUT is at most equal to the number of states in the specification.

The traditional testing theory for LTS stems from the work on testing equivalences [DNH84]: Canonical Tester, CO-OP, etc., but these use the interaction-based communication scheme as explained above. More recent are test generation algorithms that use the distinction between inputs and outputs [Pha94, Tre95].

A problem in almost all algorithms is how to deal with data parameters in interactions, especially if the existence of transitions depends on predicates over these data. Only first, rather ad-hoc solutions are currently available.

Normative choices for test generation algorithms need, and should not be made, as they follow from choices for specification and modelling formalisms, an implementation relation, and an test architecture. In principle, they are the freedom of test-suite developers or test-tool developers. However, it would help if guidance were provided in how to develop test generation algorithms for the limited set of predefined implementation relations, and in how to show their soundness.

To compare the quality of test suites FMCT proposes to define a coverage function with, again, minimal requirements. In order to do meaningful comparison of test suites, based on well-established and standardized criteria, coverage measures need to be standardized. However, this appears to be a difficult issue, and active research is going on in this direction, which makes it difficult to establish consensus on a suitable definition of such a measure now. Heuristic approaches are expected to prevail for the time being.

Acknowledgements

The contents of this paper are not just the work of the three authors. The ideas were taken from the FMCT working document [ISO96] which has been developed by the joint ITU/ISO project on FMCT. Numerous people were and are still involved in this work. In particular we would like to mention Ana Cavalli, Anne Rouger, Jan Ellsberger, Jean–Philippe Favreau, Lex Heerink, Pim Kars, Finn Kristoffersen, Jan Kroon, Marc Phalippou, Tomas Robles and Ümit Uyar who built the core of the project and supplied major contributions to the current version of the document.

References

[Abr87] S. Abramsky. Observational equivalence as a testing equivalence. *Theoretical Computer Science*, 53(3):225–241, 1987.

[DNH84] R. De Nicola and M.C.B. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.

[FH95] L.M.G. Feijs and M. Huizer. TSF: A test specification formalism. In A. Ponse, C. Verhoef, and S.F.M. van Vlijmen, editors, *Proceedings: ACP'95*, number 95-14 in Computing Science Report, pages 347–365, Eindhoven, The Netherlands, 1995. Eindhoven University of Technology.

[Gla93] R.J. van Glabbeek. The linear time – branching time spectrum II (The semantics of sequential systems with silent moves). In E. Best, editor, *CONCUR'93*, Lecture Notes in Computer Science 715, pages 66–81. Springer-Verlag, 1993.

[Hog92] D. Hogrefe. OSI formal specification case study: The INRES protocol and service, revised. Technical Report IAM-91-012, Update 1992, Universität Bern, Institut für Informatik und Angewandte Mathematik, Bern, Switzerland, 1992.

[Hog95] D. Hogrefe. Framework for formal methods in conformance testing. In T. Mizuno, T. Higashino, and Shiratori. N., editors, *Seventh Int. Workshop on Protocol Test Systems*, IFIP Transactions. Chapman & Hall, 1995.

[HT96] L. Heerink and J. Tretmans. Formal methods in conformance testing: A probabilistic refinement. In B. Baumgarten, H.-J. Burkhardt, and A. Giessler, editors, *Ninth Int. Workshop on Testing of Communicating Systems*, pages 261–276. Chapman & Hall, 1996.

[ISO90] ISO TC97/SC21. Guidelines for the application of Estelle, LOTOS and SDL. technical report TR 10167, ISO, 1990.

[ISO91] ISO. *Information Technology, Open Systems Interconnection, Conformance Testing Methodology and Framework*. International Standard IS-9646. ISO, Geneve, 1991. Also: CCITT X.290–X.294.

[ISO95] ISO/IEC JTC1/SC21 WG7, ITU-T SG 10/Q.8. *Information Retrieval, Transfer and Management for OSI, FMCT Guidelines on Test Generation Methods from Formal Descriptions, working draft*. Committee Draft CD 13245-2. ISO, February 1995. Annex to [ISO96].

[ISO96] ISO/IEC JTC1/SC21 WG7, ITU-T SG 10/Q.8. *Information Retrieval, Transfer and Management for OSI; Framework: Formal Methods in Conformance Testing*. Committee Draft CD 13245-1, ITU-T proposed recommendation Z.500. ISO – ITU-T, Geneve, 1996.

[Pha94] M. Phalippou. *Relations d'Implantation et Hypothèses de Test sur des Automates à Entrées et Sorties*. PhD thesis, L'Université de Bordeaux I, France, 1994.

[Tre95] J. Tretmans. Testing labelled transition systems with inputs and outputs. In A. Cavalli and S. Budkowski, editors, *Participants Proceedings of the Int. Workshop on Protocol Test Systems VIII — COST 247 Session*, pages 461–476, Evry, France, September 4-6 1995. Institut National des Télécommunications. Extended abstract of Memorandum INF-95-26, University of Twente, Enschede, The Netherlands, 1995.

[Vaa91] F. Vaandrager. On the relationship between process algebra and Input/Output Automata. In *Logic in Computer Science*, pages 387–398. Sixth Annual IEEE Symposium, IEEE Computer Society Press, 1991.

[WP96] T. Walter and B. Plattner. Prospect – a proposal for a new test specification language and its implementation. In A. Cavalli and S. Budkowski, editors, *Eight Int. Workshop on Protocol Test Systems*. Chapman & Hall, 1996.